Scanner User Manual

Contents

1. Introduction
    - 1.1 Scanner Program
    - 1.2 Operation Introductory
    - 1.2.1 Keystroke Mode
    - 1.2.2 Message Mode
2. Options
    - 2.1 Display Tray Icon
    - 2.2 Setting Gun Mode
    - 2.3 Running and Stopping Scanner
    - 2.4 Barcode Setup
    - 2.5 Character String Options
    - 2.6 Image Capture
    - 2.7 Image Setup
3. Program Tutorial
    - 3.1 Library Preparation
    - 3.2 Barcode Reading in the Keystroke Mode
    - 3.3 Barcode Reading in the Message Mode
    - 3.4 Trigger Key Emulation
    - 3.5 Barcode Setup
4. C++ Program Tutorial
    - 4.1 Library Preparation
    - 4.2 Barcode Reading in Keystroke Mode
    - 4.3 Trigger Key Emulation
    - 4.4 Barcode Setup
5. Visual Basic.Net Program Tutorial
    - 5.1 Library Preparation
    - 5.2 Barcode Reading in Keystroke Mode
    - 5.3 Trigger Key Emulation
    - 5.4 Barcode Setup
6. Reference

1. Introduction

1.1 Scanner Program

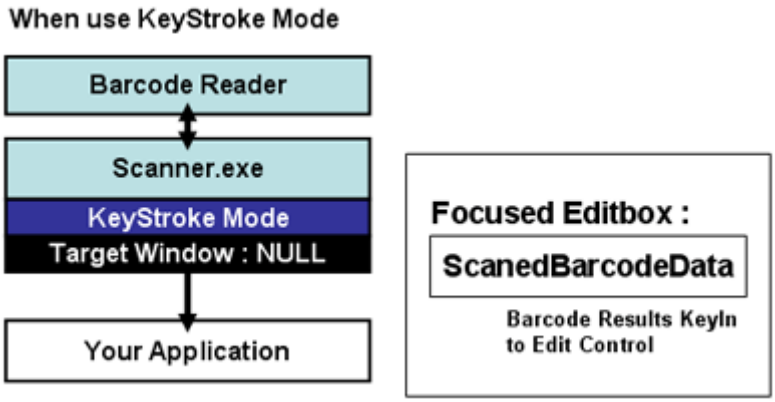
The OptimusPDA is available with 3 different scan engines – 1D laser, 1D CMOS and 2D Imager. The scanner program works with different types of decoder modules and is compatible with all three barcode scan engines available.

If a developed application is supported by the ***Scanner Program*** then it can also be used with other barcode scan engines as well. Moreover, in case when an application program is based on the **Scanner Program,** it can be used in other working environments.

1.2 Operation Introductory

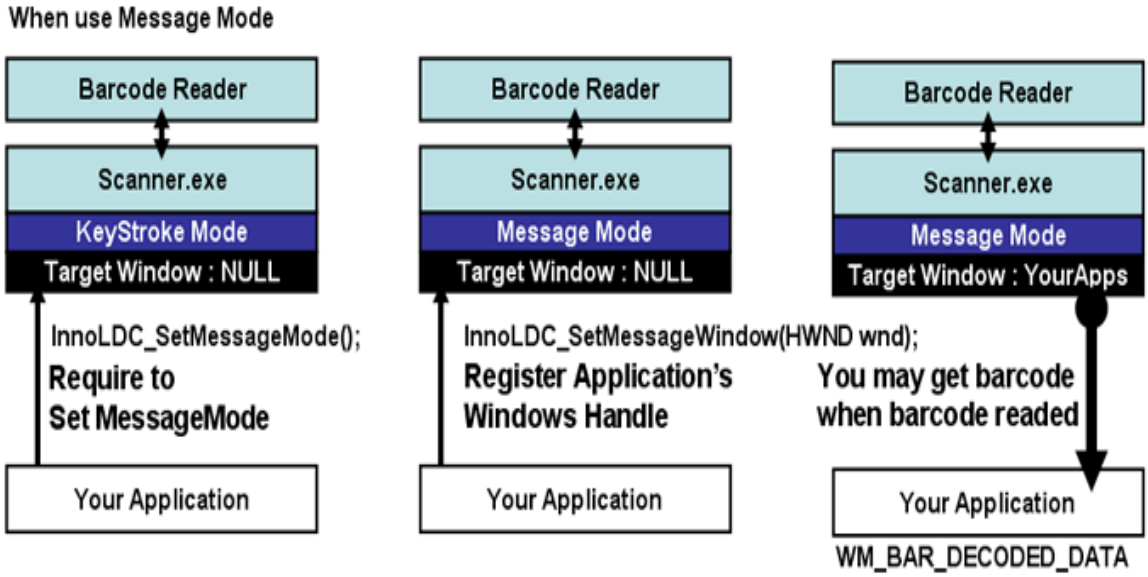
1.2.1 KeyStroke Mode

The **KeyStroke Mode** allows scanning without the need to use an additional application. Scanning information is displayed in the **Editbox** when the *"Scanner.exe"* decodes a barcode using the activated edit control tool. The **Editbox** allows decoded barcode data to be observed.



1.2.2 Message Mode

Message Mode decodes barcodes using "*Scanner.exe"* and sends a message such as WM_BAR_DECODED_DATA to the application that is being used. Therefore, decoded data is directly processed by the application. It is recommended to set up **a Window Handle**, which will receive the data.

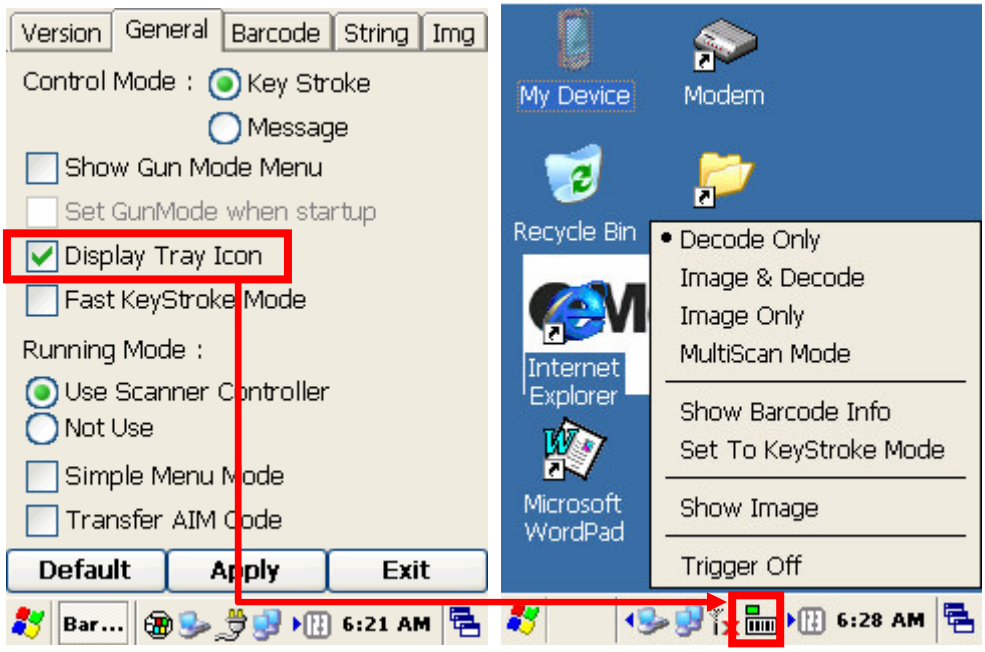


2. Options

2.1 Display Tray Icon

Quick access to the file *Scanner.exe* performance can be achieved by placing the **Tray Icon** in the taskbar.

To display this **Tray Icon**: Select the Start Tab -> Settings-> Control Panel -> Barcode.



After marking the checkbox and clicking the **OK** button, the program will put the **Tray Icon** in the taskbar at the bottom of the screen.

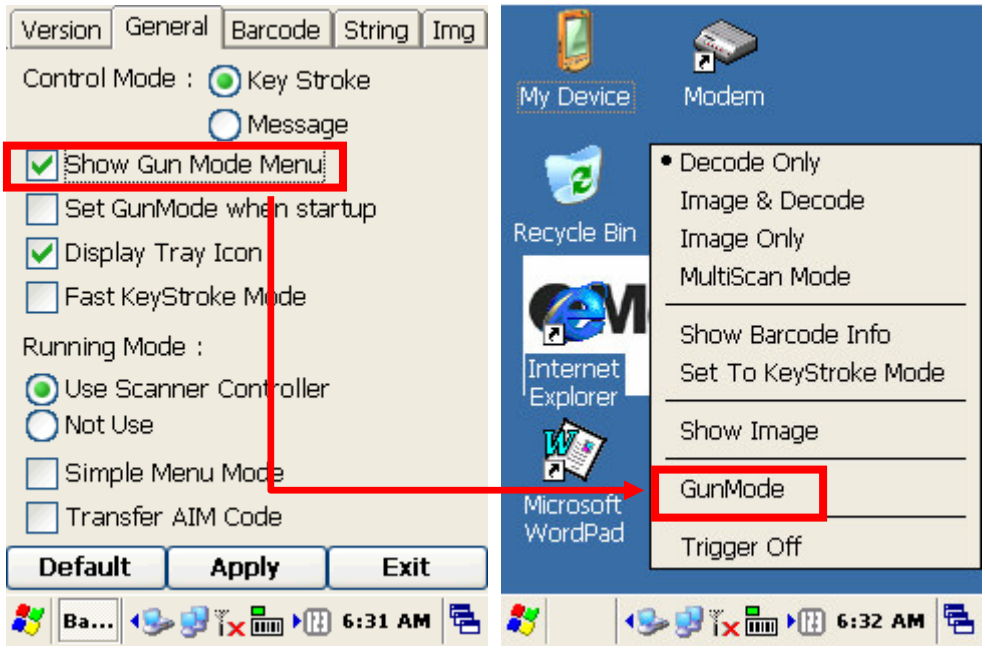
2.2 Setting Gun Mode

When the **Gun Mode** is set up, only the PDA's Gun trigger can to be used for scanning. Please note that with **Gun Mode** activated, the ActiveSync application becomes inactive. Gun Mode would need to be disable again to use the OptimusPDA with ActiveSync.

Setting the **Gun Mode**

* In the Barcode setting (in Control Panel), tick the **Use Gun Mode option** and then press the **OK** button.

* When you click the barcode **Tray Icon**, select **Gun Mode**, the **Tray Icon** then toggles



2.3 Running and stopping the scanner

To run or stop the scanner program use the Barcode Setting window according to the following methods:

- Scan Run: Settings>Barcode>General Tab, Click Exit Scanner Button.
- Scan Stop: Settings>Barcode>General Tab, Click Run Scanner Button.

Scanner program should be used only with regard to specific purposes, there is no need to keep stopping and running the scanner program.

2.4 Barcode Setup

Setting can be changed for individual barcodes. These changes can be made in the following location:

Setting >Barcode>Barcode Tab: Choose a barcode type you want to use.

2.5 Character String Options

While decoding the **Barcode** in **KeyStroke Mode**, the output can have a character string added. Below are the options:

*None: After barcode output - no input.

*Carriage Return: Enter Carriage Return input after barcode input.

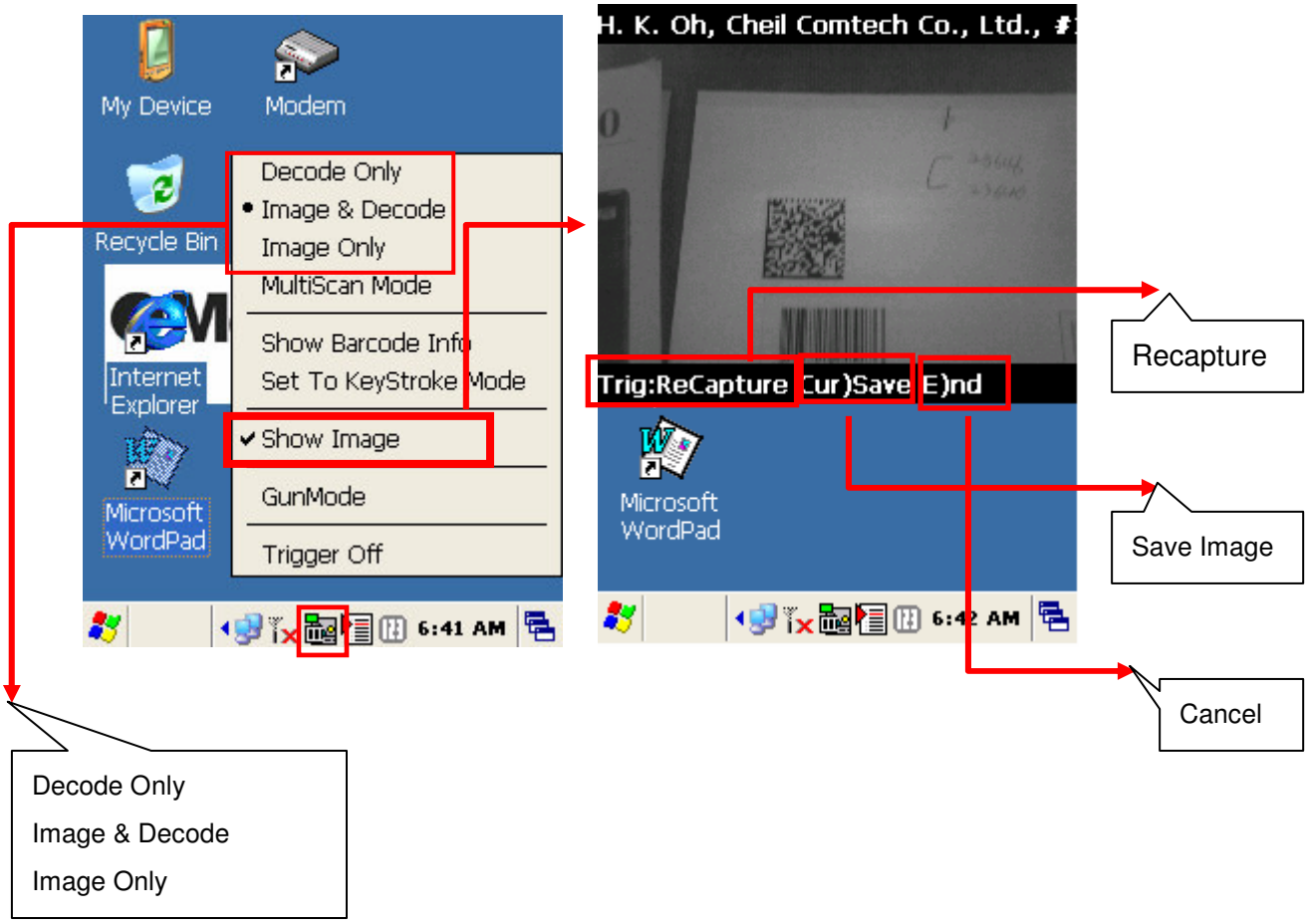
*Tab: After displayed barcode character string output - input a tab.

*Prefix String: Character string is displayed before the **Barcode Data**.

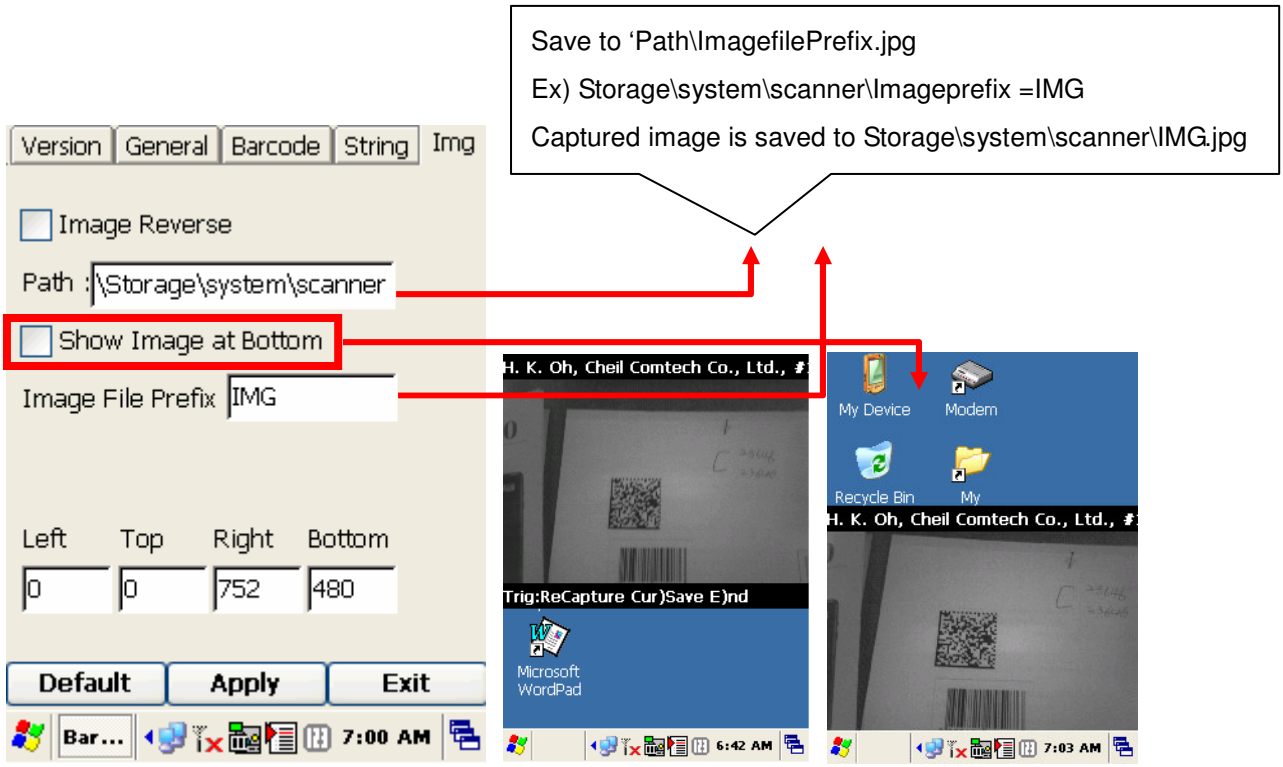
*Suffix String: Character string follows the **Barcode Data**.

EX) Prefix String ->'['Setting, Suffix String->']' **Setting Barcode Data**: When character string is 88073670649 - output data will display "[88073670649]"

2.6 Image Capture



2.7 Image Setup

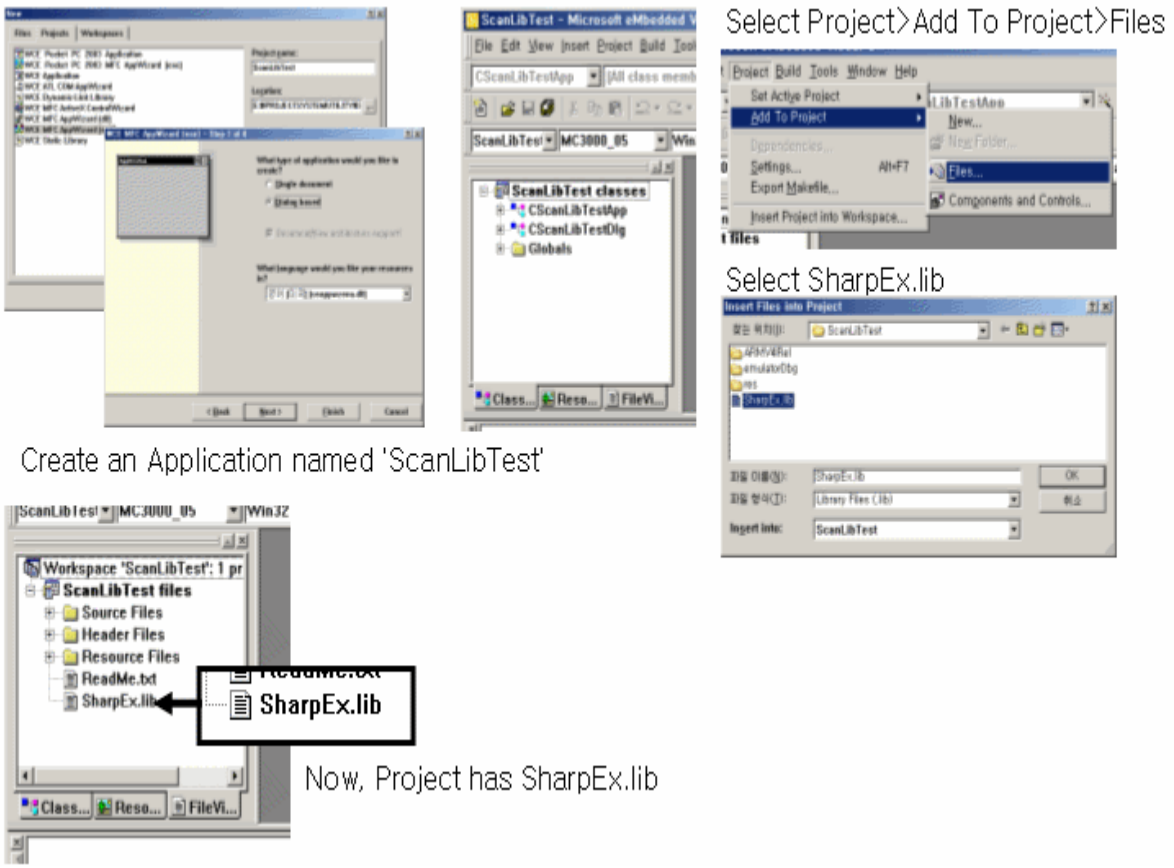


3. Program Tutorial

The principal task for the **Scanner program** is to detect when the **Trigger Key** has been activated and then send a trigger signal to scan the **Barcode** and decode. When the decoder executes a task of decoding a barcode, it works in accordance with the operation mode. Most importantly, it sends every decoded result to the application as a type of **KeyStroke** or **Window Message** designated for further processing.

3.1 Library Preparation

If there is no library, it is reasonable to use the **KeyStroke Mode** function or employ WM_BARCODE_DATA. In such cases the DLL is not required. As soon as functions are used that support capturing a barcode, the DLL is needed. An illustration on how to use library is presented below.



```
#include "BarcodeTypeHHP.h"
#include "BarcodeTypeIntermac.h"
#include "BarcodeTypeSymbol.h"
#include "SharpEx.h"              // Use SharpEx Lib.
```

After opening the library, a header file should be added as in the above source code.

3.2 Barcode Reading in Keystroke Mode

A program assigned to the **Focus** in the **Editbox** decodes a barcode and forwards the **Barcode Data** to the **Editbox**.

Highlights on the source code in the **KeyStroke Mode:**

- 1. To switch **KeyStroke Mode** on: **SetOpMode** (RET_KEYSTROKE);
- 2. As soon as the PDA's **Scan Trigger** key is being pressed, F20 key value is inputted to the O/S.

Now a **SetFocus** can be specified for the **Editbox** where a **PreTranslateMessage** (MSG* pMsg) is stored.

```
BOOL CScanLibTestDlg::PreTranslateMessage(MSG* pMsg)
{
    if ( pMsg->message == WM_KEYDOWN ) {
        if ( pMsg->wParam == VK_F20 ) // When Trigger Key Pressed
        {
            CEdit * edtBarcode = (CEdit*)GetDlgItem(IDC_EDIT_BARCODE);
            edtBarcode->SetFocus();
        }
    }
    return CDialog::PreTranslateMessage(pMsg);
}
...
```

Transfer of readings with regard to 2D barcode working under the KeyStroke Mode (IT50x0 only)

The 2D barcode content includes a character string and it is important that the **BinaryData** is assigned to the character strings. There is an alternative option that makes the **KeyMessage** workable when the **Message Mode** cannot be used. By setting the appropriate **KeyEvent** setting, which occurs when capturing 2D Barcode (F15), you can see the **Barcode** content through the **GetLastBarcode** function.

```
BOOL CScanLibTestDlg::PreTranslateMessage(MSG* pMsg)
{
    if ( pMsg->message == WM_KEYDOWN ) {
        if ( pMsg->wParam == VK_F15 )
        {   // When Decoded 2D Barcode
            TCHAR Temp[64];
            GetLastBarcode( Temp ); // Get 2D Barcode
            AfxMessageBox( Temp );
        }
    }
    return CDialog::PreTranslateMessage(pMsg);
}
```

3.3 Barcode Reading in Message Mode

The executable file scanner.exe that decodes **Barcode** allows sending information (WM_BAR_DECODED_DATA) to the application as well as receiving a direct response.

The **Window Handle**, which is responsible for receiving messages, must be identified in a proper way, so the beam will work effectively in order to decode a **Barcode**.

Explanatory tips for the source code in the **Message Mode:**

- 1. The function for changing the **Message Mode** is SetOpMode (RET_MESSAGE);
- 2. When you are working in the **Message Mode**, you should assign the **Window Handle** to the application.

* Notice
*Beam does not work unless **Window Handle** is being assigned.*

(SetMessageWindow(this->m_hWnd)

```
BOOL CScanLibTestDlg::OnInitDialog()
{
    ... omission ...

    SetOpMode( RET_MESSAGE );                  // Change to MessageMode
    SetMessageWindow( this->m_hWnd );      // Set Windows Target to Current Window

    ... omission ...
}
```

- 3. If the scanner works in a proper way during decoding phase, it sends a message (WM_BAR_DECODED_DATA) to the **OnDecodeData() Function** in order to process received **Barcode Data**.

Adding a function to the header file

```
    ... omission ...
    afx_msg void OnDecodeData(WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP()
};
```

Adding a message handler to the source code.

```
BEGIN_MESSAGE_MAP(CScanLibTestDlg, CDialog)
... omission ...
      //}}AFX_MSG_MAP
    ON_MESSAGE(WM_BAR_DECODED_DATA, OnDecodeData)
END_MESSAGE_MAP()

... omission ...

void CScanLibTestDlg::OnDecodeData(WPARAM wParam, LPARAM lParam)
{
    TCHAR    Temp[64];
    ::GetWindowText( (HWND)wParam, Temp, 64); // Temp 변수에  Barcode Data 를  저장

    AfxMessageBox( Temp );
}
```

3.4 Trigger Key Emulation

In some circumstances it may be necessary to use a different key than the **Trigger Keys** located on both sides of the PDA. In these cases activation of the **Trigger Key** emulation function is highly recommended.

- void TrigKeyDownEmu();
- void TrigKeyUpEmu();

The above indicated two functions are created to switch program into a user's required operating mode. After accomplishing this task you can press the **Trigger Key** on your PDA.

If there is a need to use the cursor key located at the center of your PDA as a trigger key, you should add this function to the program as it is illustrated below.

```
BOOL CScanLibTestDlg::PreTranslateMessage(MSG* pMsg)
{
   if ( pMsg->message == WM_KEYDOWN ) {
   if (pMsg->wParam == VK_UP ||
       pMsg->wParam == VK_DOWN ||
       pMsg->wParam == VK_LEFT ||   // When Pressed Cursor Key
       pMsg->wParam == VK_RIGHT ) TrigKeyDownEmu();
   }
   else if ( pMsg->message == WM_KEYUP ) {
   {
   if (pMsg->wParam == VK_UP ||
       pMsg->wParam == VK_DOWN ||
       pMsg->wParam == VK_LEFT ||   // When Released Cursor Key
       pMsg->wParam == VK_RIGHT ) TrigKeyUpEmu();
   }
   return CDialog::PreTranslateMessage(pMsg);
}
```

3.5 Barcode Setup

The **Barcode Type** selection can be done from the **Control Panel.** However it is also possible to send a direct request to the *Scanner.exe* file to set the **Barcode Type**.

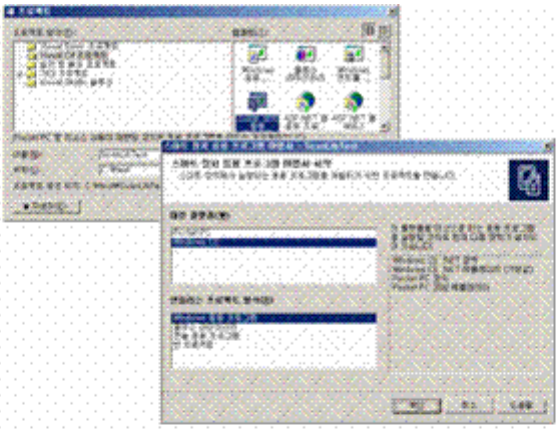
```
void CScanLibTestDlg::OnButtonCode39Enable()
{
   SetBarcode(BAR_CODE39, TRUE);
}

void CScanLibTestDlg::OnButtonCode39Disable()
{
   SetBarcode(BAR_CODE39, FALSE);
}
```

Bar Code Type:

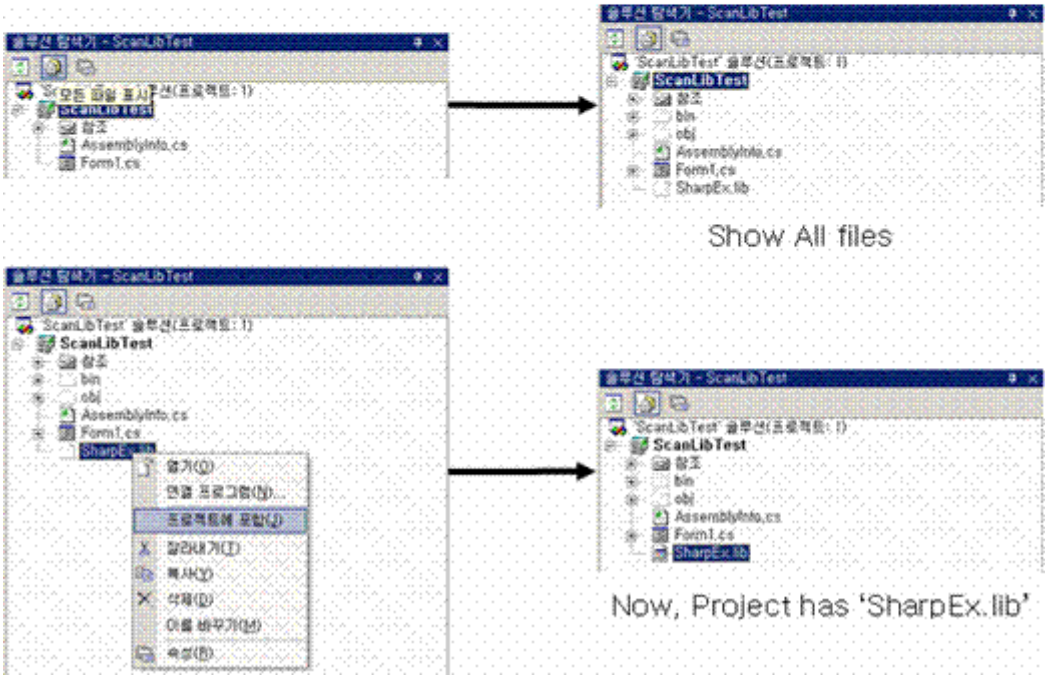
BAR_PDF417	0
BAR_CODE39	1
BAR_CODABAR	2
BAR_CODE128	3
BAR_DISCRETE2OF5	4
BAR_INTERLEAVED2OF5	5
BAR_CODE93	6
BAR_UPC_A	7
BAR_UPC_E0	8
BAR_EAN8	9
BAR_EAN13	10
BAR_MSIPLESSY	11
BAR_EAN128	12
BAR_UPC_E1	13
BARCODE_NUMBER	14
BAR_NONE	98
BAR_REGONLY	99

4. C# Program Tutorial

4.1 Library Preparation



Create an Application named 'ScanLibTest'



After activating library, the source code will be shown:

```
[DllImport("SharpEx.dll")]
```

```
private static extern bool SetOpMode(int OpMode);
[DllImport("SharpEx.dll")]
private static extern bool GetLastBarcode(char[] Buffer);
[DllImport("SharpEx.dll")]
private static extern void TrigKeyUpEmu();
[DllImport("SharpEx.dll")]
private static extern void TrigKeyDownEmu();
[DllImport("SharpEx.dll")]
private static extern bool setBarCode(int Type, bool bEnable);
```

4.2 Barcode Reading in Keystroke Mode

In cases when the application runs under the **Focus** status in the **Editbox** and without selection of any specific coding program, it will decode a barcode by default to display **Barcode Data** in the **Editbox**.

Highlights on the source code in the KeyStroke Mode:

1. To switch **KeyStroke Mode** on: **SetOpMode(0);**
2. When **Scan Trigger** key of the PDA is pressed, F20 Key value will be inputted to the O/S

Now, **SetFocus** can be specified in the **Editbox** where **OnKeyDown** (object sender, System. Windows. Forms. KeyEventArgse) should be located.

```
private void OnKeyDown(object sender, System.Windows.Forms.KeyEventArgs e)
{
    if (e.KeyCode == Keys.F20)
    {
        edtBarcode.Focus();
        edtBarcode.SelectAll();
    }
}
```

Transfer of readings with regard to 2D barcode working in KeyStroke mode (IT40x0 only)

Usually 2D barcode content contains character string along with **BinaryData**.

The 2D barcode content includes a character string and it is important that the **BinaryData** is assigned to the character strings. There is an alternative option that makes the **KeyMessage** workable when the **Message Mode** cannot be used. By setting the appropriate **KeyEvent** setting, which occurs when capturing 2D Barcode (F15), you can see the **Barcode** content through the **GetLastBarcode** function.

```
private void OnKeyDown(object sender, System.Windows.Forms.KeyEventArgs e)
{
    if (e.KeyCode == Keys.F15)
    {
        char[] temp = new char[64];
        GetLastBarCode(temp);
        BarCode.Text = temp;
    }
}
```

4.3 Trigger Key Emulation

In some circumstances it may be necessary to use a different key than the **Trigger Keys** located on both sides of the PDA. In these cases activation of the **Trigger Key** emulation function is highly recommended.

- void TrigKeyDownEmu();
- void TrigKeyUpEmu();

The above indicated functions are created to switch program into a user's required operating mode. After accomplishing this task you can press the **Trigger Key** on your PDA.

If there is a need to use the cursor key located at the center of your PDA as a trigger key, you should add this function to the program as it is illustrated below.

```
private void OnKeyDown(object sender, System.Windows.Forms.KeyEventArgs e)
{
```

```
    if (e.KeyCode == Keys.Up ||
        e.KeyCode == Keys.Down ||
        e.KeyCode == Keys.Left ||
        e.KeyCode == Keys.Right)
    {
        TrigKeyDownEmu();
    }
}

private void OnKeyUp(object sender, System.Windows.Forms.KeyEventArgs e)
{
    if (e.KeyCode == Keys.Up ||
        e.KeyCode == Keys.Down ||
        e.KeyCode == Keys.Left ||
        e.KeyCode == Keys.Right)
    {
        TrigKeyUpEmu();
    }
}
```

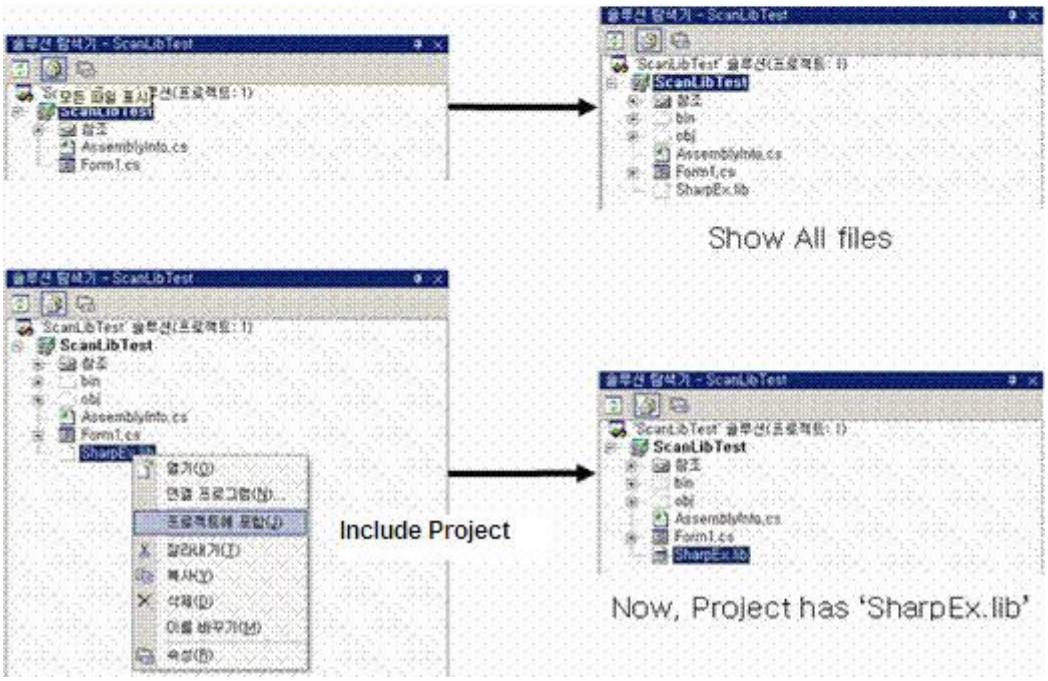
4.4 Barcode Setup

To change or set up different setting for a particular **Barcode Type**, modifications should be made through the control panel or choose a direct execution by activating the Scanner.exe file.

```
private void Code39Enable_Click(object sender, System.EventArgs e)
{
    SetBarcode(1, true);
}

private void Code39Disable_Click(object sender, System.EventArgs e)
{
    SetBarcode(1, false);
}
```

5. Visual Basic.Net Program Tutorial

5.1 Library Preparation



For included library the following source code should be applied.

```
<DllImport("SharpEx.dll")> _
Public Shared Function SetOpMode(ByVal OpMode As Int32) As Boolean
End Function
<DllImport("SharpEx.dll")> _
Public Shared Function GetLastBarcode(ByVal buffer() As Char) As Boolean
End Function
<DllImport("SharpEx.dll")> _
Public Shared Sub TrigKeyUpEmu()
End Sub
<DllImport("SharpEx.dll")> _
Public Shared Sub TrigKeyDownEmu()
End Sub
<DllImport("SharpEx.dll")> _
Public Shared Function SetBarcode(ByVal Type As Int32, ByVal bEnable As Boolean) As Boolean
End Function
```

5.2 Barcode Reading in Keystroke Mode

When the application is under the **Focus** status in the **Editbox** and without selection of any specific decoding program, it will decode a barcode by default to display **Barcode Data** in the **Editbox**.

Highlights on the source code in the KeyStroke Mode

1. To switch **KeyStroke Mode** on: **SetOpMode(RET_KEYSTROKE);**

2. When **Scan Trigger** key of the PDA is pressed, F20 Key value will be inputted to the O/S.

Then you can specify a **SetFocus** in the **Editbox** where Form1_KeyDown(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs) Handles MyBase.KeyDown and edtBarcode_KeyDown(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs) Handles edtBarcode.KeyDown should be located.

```
Private Sub Form1_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
 Handles MyBase.KeyDown
    If (e.KeyCode = Keys.F20) Then
        edtBarcode.Focus()
        edtBarcode.SelectAll()
    End If
End Sub

Private Sub edtBarcode_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
 Handles edtBarcode.KeyDown
    If (e.KeyCode = Keys.F20) Then
        edtBarcode.Focus()
        edtBarcode.SelectAll()
    End If
End Sub
```

Transfer of readings with regard to 2D barcode working in KeyStroke mode (IT40x0 Only)

Any 2D barcode's contents contain a string of characters associated with **BinaryData.**

There is an alternative option that makes the **KeyMessage** workable when the **Message Mode** cannot be used. By setting the appropriate **KeyEvent** setting, which occurs when capturing 2D Barcode (F15), you can see the **Barcode** content through the **GetLastBarcode** function.

```
Dim temp() As Char
Dim strBarcode As String

Private Sub Form1_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
  Handles MyBase.KeyDown
      If (e.KeyCode = Keys.F15) Then
            GetLastBarcode(temp)
            strBarcode = charTostring(temp)
            edtBarcode.Text = strBarcode
      End If
End Sub

Private Sub edtBarcode_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
  Handles edtBarcode.KeyDown
      If (e.KeyCode = Keys.F15) Then
            GetLastBarcode(temp)
            strBarcode = charTostring(temp)
            edtBarcode.Text = strBarcode
      End If
End Sub

Private Function charTostring(ByVal temp() As Char) As String
      Dim i As Integer
      Dim strTemp As String

      strTemp = ""
      For i = 0 To UBound(temp) - 1
            strTemp = strTemp + temp(i)
      Next i

      charTostring = strTemp
End Function
```

5.3 Trigger Key Emulation

In some circumstances it may be necessary to use a different key than the **Trigger Keys** located on both sides of the PDA. In these cases activation of the **Trigger Key** emulation function is highly recommended.

Below specified functions are created to switch the program into an operating mode required by a user. After choosing relevant function you can press the **Trigger Key** located on your PDA.

- void TrigKeyDownEmu();
- void TrigKeyUpEmu();

If there is a need to employ the cursor key located at the center of OptimusPDA as a trigger key, you should use the following function of the program.

```
Private Sub Form1_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
 Handles MyBase.KeyDown
    If e.KeyCode = Keys.Up Or e.KeyCode = Keys.Down Or e.KeyCode = Keys.Left Or e.KeyCode =
Keys.Right Then
        TrigKeyDownEmu()
    End If
End Sub

Private Sub Form1_KeyUp(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs)
 Handles MyBase.KeyUp
    If e.KeyCode = Keys.Up Or e.KeyCode = Keys.Down Or e.KeyCode = Keys.Left Or e.KeyCode =
Keys.Right Then
        TrigKeyUpEmu()
    End If
End Sub
```

```
Private Sub edtBarcode_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
 Handles edtBarcode.KeyDown
    If e.KeyCode = Keys.Up Or e.KeyCode = Keys.Down Or e.KeyCode = Keys.Left Or e.KeyCode =
Keys.Right Then
        TrigKeyDownEmu()
    End If
End Sub

Private Sub edtBarcode_KeyUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs)
 Handles edtBarcode.KeyUp
    If e.KeyCode = Keys.Up Or e.KeyCode = Keys.Down Or e.KeyCode = Keys.Left Or e.KeyCode =
Keys.Right Then
        TrigKeyUpEmu()
    End If
End Sub
```

5.4 Barcode Setup

To change or set up different setting for a particular **Barcode Type**, modifications should be made through the control panel or choose a direct execution by activating the Scanner.exe file.

```
Private Sub Code39Enable_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
 Handles Code39Enable.Click
    SetBarcode(1, True)
End Sub

Private Sub Code39Disable_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
 Handles Code39Disable.Click
    SetBarcode(1, False)
End Sub
```

6. Reference

BOOL GetLastBarcode(LPSTR Barcode);

Loading decoded result to the LPSTR pointer

Parameter

Barcode

[out] sending a pointer that will receive a barcode

Return

If the scanner software is involved in processing data, return is TRUE.

If the scanner software is not involved in processing data, return is FALSE.

BOOL SetOpMode(int OpMode);

Decode mode change

Parameter

OpMode

RET_KEYSTROKE: Switch to the KeyStroke Mode.

RET_MESSAGE: Switch to the Message Mode.

Return

If the scanner software is involved in processing data, return is TRUE.

If the scanner software is not involved in processing data, return is FALSE.

void TrigKeyDownEmu();

Emulation when the side **Trigger Key is** being pressed

void TrigKeyUpEmu();

Emulation when the side **Trigger Key is** being released

BOOL SetMessageWindow(HWND wnd);

Parameter

wnd

Transfer result from the handle to the window that processes the barcode data.

Return

If the scanner software is involved in processing data, return is TRUE.

If the scanner software is not involved in processing data, return is FALSE.

BOOL SetBarcode(int Type, BOOL bEnable);

Parameter

Type

Select a barcode you wish to setup. A barcode has several types that enable decoding. Depending on decoder's program you should refer to the relevant file header that corresponds to it.

File headers are classified by decoder according to the following types:

- BarcodeTypeHHP.h : HHP IT50x0
- BarcodeTypeSymbol.h : Symbol SE955/SE923/SE923HS
- BarcodeTypeIntermac : EV15

Decide which option **(Barcode Enable** or **Barcode Disable)** should be selected depending on its compatibility to the certain barcode type.

Return

If the scanner software is involved in processing data, returns is TRUE.

If the scanner software is not involved in processing data, return is FALSE.